MITSloan
Management Review

Georg von Krogh

Open-Source Software Development

REPRINT NUMBER 4432

RESEARCH BRIEF

Open-Source Software Development

An overview of new research on innovators' incentives and the innovation process. **by Georg von Krogh**

Open-source software development projects — Internet-based communities of software developers who voluntarily collaborate in order to develop software that they or their organizations need — have become an important economic and cultural phenomenon. Sourceforge.net, a major infrastructure provider and repository for such projects, lists more than 10,000 of them and more than 300,000 registered users. The digital software products emanating from such projects are commercially attractive and widely used in business and government (by IBM, NASA and the German government, to name just a few). Because such products are deemed a "public good" — meaning that one person's use of them does not diminish another's benefits from them — the open-source movement's unique development practices are challenging the traditional views of how innovation *should* work.

A Brief History

In the 1960s and 1970s, software development was carried out mostly by scientists and engineers working in academic, government and corporate laboratories. They considered it a normal part of their research culture to freely exchange, modify and build upon one another's software, both individually and collaboratively. In 1969, the U.S. Defense Advanced Research Projects Agency (DARPA) established the ARPAnet, the first transcontinental, high-speed computer network. ARPAnet allowed developers to exchange software code and other information widely, easily, swiftly and cheaply. It grew in popularity and eventually linked several universities, defense contractors and research laboratories. However, its limits soon became apparent. The network could connect approximately 250 hosts, too few to cater to the growing communication needs among engineers and academics. A number of technological advancements that emerged between 1940 and 1970 led to the development of the Internet project that would eventually solve this bottleneck. Today the Internet has more than 100 million users worldwide and has become the major breeding ground for open-source software development.

The communal culture was strongly present among a group of programmers at the MIT Artificial Intelligence Laboratory in the 1960s and 1970s. In the 1980s, this group received a major jolt when MIT licensed some of the code to a commercial software firm, which promptly restricted access to the source code of that software, and hence prevented noncompany personnel — including MIT hackers who had participated in developing it — from continuing to use it as a platform for further learning and development.

Richard Stallman, an accomplished programmer at the Artificial Intelligence Laboratory, was rather distraught and somewhat offended by this loss of access to communally developed code, and he lamented a general trend in the software world towards the development of proprietary packages that could not be studied or modified by others. In 1985, he founded the Free Software Foundation with the intention to develop and diffuse a legal mechanism that would allow developers to preserve the "free" status of their software by using their own copyright to grant software licenses that would guarantee a number of rights to *all* future users. The basic license developed by Stallman, in order to implement this idea, was the General Public License or GPL. The basic rights transferred to those possessing a copy of free software included the right to use it at no cost, the right to study and modify its "source code," and the right to distribute modified or unmodified versions to others at no cost.

The free software idea did not immediately become mainstream; industry was actually rather suspicious of it. For example, firms feared the possible "viral effects" of the GPL license, meaning that, after software in the GPL regime is combined with proprietary software, it would prove difficult to restrict user access through normal licensing or controlling the source code. In 1998, Bruce Perens and Eric Raymond agreed that a significant part of the problem resided in Stallman's term "free" software. The term might understandably have an ominous ring to it when heard by individuals in the business world. Accordingly, they, along with other prominent hackers, founded the "open source" software movement. Open-source software incorporates essentially the same licensing practices as those pioneered by the free software movement, covering free redistribution of software and the inclusion of the source code of a program. These licensing

Professor Georg von Krogh *is director of the Institute of Management, and cofounder of the research center KnowledgeSource, at the University of St. Gallen in Switzerland. Contact him at georg.vonkrogh@unisg.ch.*

practices also apply to derived works in that the rights attached to the original program apply to all who build upon the source code, without these programmers needing to provide additional licenses.

Incentives To Innovate

Under the aegis of open-source licensing practices, which guarantee that the products cannot be withheld from anyone's use, what are the incentives to innovate, and, given that most open-source projects exist outside the firm's boundaries, how does this innovation process work?

People and firms innovate because there are private incentives to do so. For example, entrepreneurs use their own funds to develop knowledge and products that generate revenue streams (and employees are paid for their creative services to the company). Society also encourages innovation by putting in place mechanisms to protect intellectual property associated with products, so that future revenue streams can be guaranteed for the innovator.

However, a central question raised by the success of open-source software development has been succinctly stated by two economists, Josh Lerner and Jean Tirole: "Why should thousands of top-notch programmers contribute freely to the provision of a public good?" Open-source software developers are rarely paid for their services, and the licenses and hacker practices make it difficult, if not impossible, for these developers to appropriate returns from their products. Eric von Hippel and I suggest that open-source software developers freely reveal and share because they garner personal benefits from doing so, such as learning to develop complex software, perfecting expertise with a computer language, enhancing their reputation, and for pure fun and enjoyment. Many of these benefits depend on membership in a well-functioning developer community. Typically, in open-source communities, members give direct, specific and immediate feedback on the software code that others write and submit. This peer-review process is not only valuable for the individual who

Referenced Research

J. Lerner and J. Tirole, "The Simple Economics of Open Source" NBER working paper no. w7600 (Cambridge, Massachusetts: National Bureau of Economic Research, March 2000)
A foundational paper for research on open-source software, arguing that individuals contribute to open-source software in accordance with career incentives.

E. von Hippel and G. von Krogh, "Exploring the Open Source Software Phenomenon: Issues for Organization Science" *Organization Science* 14, no. 2 (2003, in press)
The authors suggest that a combination of private and community-related benefits results from contributions to open-source software development projects.

G. Hertel, S. Niedner, and S. Herrmann, "Motivation of Software Developers in Open Source Projects: An Internet-Based Survey of Contributors to the Linux Kernel" *Research Policy* 32 (2003, in press)
The authors test two extant models from the social sciences. The first explains incentives to participate in social movements, and the second deals with motivational processes in small work teams, particularly "virtual teams." The authors report a good fit between models and data derived from a survey of 141 contributors to the Linux kernel.

N. Franke and E. von Hippel, "Satisfying Heterogeneous User Needs via Innovation Toolkits: The Case of Apache Security Software" *Research Policy* 32 (2003, in press)
The authors explore a frequently cited reason for contributing to open-source software: People innovate to better satisfy their own needs.

S. O'Mahony, "Guarding the Commons: How Open-Source Contributors Protect Their Work" *Research Policy* 32 (2003, in press)
An ethnographic study in which the author explores the various ways open-source project members encourage compliance with the terms of their project licenses.

E. von Hippel, "Economics of Product Development by Users: The Impact of 'Sticky' Local Information" *Management Science* 44 (May 1998): 629-644
Information about user needs and problems is "sticky," in the sense that it is costly to retrieve for a manufacturer. See also " 'Sticky Information' and the Locus of Problem Solving: Implications for Innovation," E. von Hippel, *Management Science* 40, no. 5 (1994): 429-439.

M.A. Cusumano, "Shifting Economies: From Craft Production to Flexible Systems and Software Factories" *Research Policy* 21, no. 5 (1992): 453-480
The author discusses the organization of large-scale, commercial software innovation.

R.D. Austin, "The Effects of Time Pressure on Quality in Software Development: An Agency Model" *Information Systems Research* 12, no. 2 (2001): 195-207
The author explores in detail the relationship between the software developer and the firm.

K. Lakhani and E. von Hippel, "How Open Source Software Works: 'Free,' User-to-User Assistance" *Research Policy* 32 (2003, in press)
How users demonstrate satisfaction and obligation in assisting each other in resolving tasks related to use of Apache software. Also see **J.Y. Moon and L. Sproull, "Essence of Distributed Work: The Case of the Linux Kernel,"** First Monday 5, no. 11 (Nov. 2000).

G. von Krogh, S. Spaeth and K. Lakhani, "Community, Joining, and Specialization in Open-Source Software Innovation: A Case Study" *Research Policy* 32 (2003, in press)
A study of growth and specialization in a community of developers.

S. Koch and G. Schneider, "Effort, Co-operation, and Co-ordination in an Open Source Software Project: GNOME" *Information Systems Journal* 12, no. 1 (2002): 27-42
Research on open-source projects from a software engineering perspective. The data are used for a first attempt to estimate the total effort to be expended on a particular project.

B. Kogut and A. Metiu, "Open-Source Software Development and Distributed Innovation" *Oxford Review of Economic Policy* 17, no. 2 (2001): 248-264
The authors of this paper argue that open-source software development is a production model that exploits the distributed intelligence of participants in Internet communities.

submits code, but also for ensuring the overall quality of the software.

The importance of community was supported in a study of contributors to the Linux operating-system kernel project. Guido Hertel, Sven Niedner and Stefanie Herrmann found that the more contributors identified themselves as "Linux developers," the higher their level of involvement and the greater their efforts to code for the project. The Linux project and communities surrounding it have the advantage of being well known, and the project's positive image can create a sense of responsibility and loyalty among developers. They also found contributors to be motivated by group-related factors such as their perceived indispensability to the team. Lastly, they found that contributors had important pragmatic motives to improve their own software. This is key to understanding why many skilled software developers work for free. Nic Franke and Eric von Hippel studied the security needs of those who use Apache Web-server security software. They found that because Apache is open-source software, users can modify it to better fit their individual needs and that users who do so are more satisfied.

OPEN-SOURCE LICENSES ARE DESIGNED TO GUARANTEE THE RIGHTS OF *FUTURE USERS* AGAINST APPROPRIATION.

While these motives explain, to a great degree, the benefits and incentives of becoming involved in open-source projects, the issue of how developer communities protect the fruits of their labor is a subject of some concern. Unlike in commercial development — where intellectual property law protects the rights of authors to appropriate economic returns from their innovations — open-source licenses are designed to guarantee the rights of *future users* against appropriation. Siobhan O'Mahony points out that open-source software contributors have an active concern that their work remains part of the commons, and they zealously protect their work to this end. Open-source project members encourage compliance with the terms of project licenses in various ways. They may exercise sanctioning via online discussions and may use brands and logos to ensure that the intellectual property they have contributed remains in the commons.

The Innovation Process

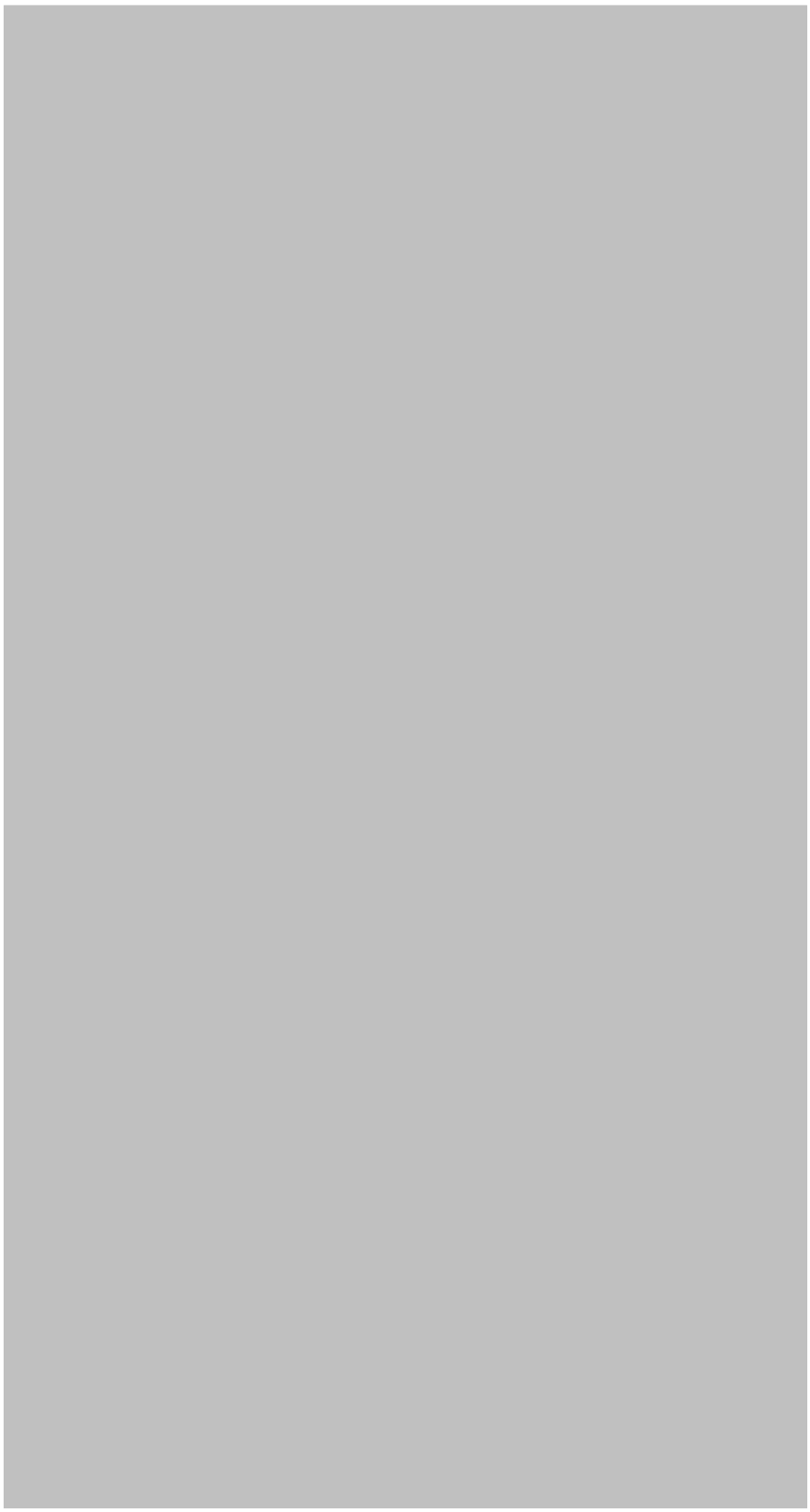
Open-source projects can be started by anyone with the appropriate programming skills and motives. Typically, an entrepreneur with some workable code or an idea for a software project launches a message on one of the many Web-based collaborative communities, such as freshmeat.net or Geocrawler.com. If others find the code useful or the idea interesting, they join in by contributing code, fixing bugs or other problems in the software, providing comments, ideas, references to other projects and so on. Over time, the number of contributors can grow substantially, and the project may use one of the many technical infrastructures available on the Web to host and monitor changes to the emerging software. The entrepreneurs also normally set up a project mailing list for posting questions and answers or messages pertinent to developing the software.

The first challenge for the open-source entrepreneur is mobilizing top-notch programmers. In some cases, when no viable commercial alternatives exist or when they are too expensive, it may, in fact, be easier to attract top programmers to open-source projects than to commercial development ventures. Developers of standard off-the-shelf software often find it difficult to judge whether a product feature will have a major impact on satisfying user needs, and they also recognize that users often have difficulty expressing their needs. Eric von Hippel has noted that such "sticky" information is costly to retrieve because understanding a user's problems requires that the manufacturer "dwell in the context of the user" for a prolonged period. As a result, the high development (and marketing) costs for standard packaged software typically cause software firms to spread these

costs among a large population of users. Companies will therefore seek information about "average user needs and problems," and this affects product development. In some cases, companies supplement this with help from external opinion leaders in order to ultimately strengthen the product design and identify bugs in early releases. For some types of software, this way of optimizing the innovation process leads to market failure and will spur the interest of developers in joining open-source software projects, in which they can code to meet their own practical and technical needs.

Another challenge for open-source entrepreneurs, as has been discussed by Michael Cusumano, is organizing the innovation process properly. For-profit programming companies often seek to reduce development costs and control quality by closely monitoring what programmers do and how they do it. To secure returns on investment in innovation, most companies try to seek out and recruit the most outstanding software talent, bind them by contract and take steps to minimize opportunism. (R.D. Austin has explored in detail the relationship between the software developer and the firm.) Additionally, software companies attempt to contain costs as well as prevent spillover of knowledge, technology and other secrets to competitors by encouraging specialization and division of labor among developers.

By contrast, the relationship between the open-source software project and its participants is largely voluntary and not regulated by formal contract. The initiator of a project might become well known in the public domain, such as Linus Torvalds who created the Linux operating-system kernel (the central module responsible for such functions as memory, process and disk management); but an initiator cannot legally force participants to continue or increase their efforts in the project. Recent work by Karim Lakhani and Eric von Hippel and by Jae Yun Moon and Lee Sproull shows that contributors to open-source software projects value a sense of ownership and control over the work



product — something they do *not* experience in programming work carried out for hire. For this reason (among others), participants of open-source software projects also do not take any particular action to minimize "free riding" (the downloading or "consumption" of remote files without reciprocal uploading or "production" of useful contributions). Project learning in the open-source world is captured in the chronology of e-mail exchanges and in the source code that is open for all to see; there is no need to contain or merge information according to a formal division of labor. By means of this transparency, a project accumulates the development efforts of volunteer users and seems to encourage the software's diffusion in order to build a developer's reputation and spread the products.

Whereas a firm may secure the best talent through the processes of professional recruiting, there is no formal recruiting in open-source software projects. This could lead to less talented individuals participating in open-source endeavors and eventually to compromises in the quality of the software. But research has shown that developer communities are meritocracies, in which technical knowledge and expertise determine a contributor's impact on the software design. Sebastian Spaeth, Karim Lakhani and I studied a sophisticated peer-to-peer software project named Freenet and found that only 30 people had the right to include code in the official version of the software. To become one of these core developers, participants had to demonstrate a considerably higher level of technical activity than other contributors. The open-source organization also self-allocates talent. As noted in our work, as well as in two other studies by Stefan Koch and Georg Schneider and by Bruce Kogut and Anca Metiu, people are not assigned tasks on the basis of predefined labor schemes, instructions and directives, but rather on the basis of interest and self-selection. This drives a high level of specialization, but implies that potentially useful software modules may never be developed.

THERE ARE GREAT ADVANTAGES TO A MODEL WHEREBY RESOURCES USED FOR INNOVATION ARE WIDELY DISTRIBUTED THROUGHOUT THE WORLD.

A View to the Future

What does the open-source software phenomenon imply for future business activities? In the short to medium term, some managers may encourage the use of open-source software in their own firms. Others may attempt to build a business based on distributing and servicing open-source software. U.S.-based Red Hat and German-based SuSE, which distribute Linux software, serve as templates for such activity. Other companies may sell computer hardware running open-source software, such as IBM, which offers Linux as an option. Managers may also try to reduce development costs and boost software standards by using the open-source software development model. Such an example is Sun Microsystems' decision to rely on open-source methods to develop and distribute Java.

The open-source software movement also provides important management lessons regarding the most effective ways to structure and implement innovation. There are potentially great advantages (and perhaps some disadvantages) to a model whereby resources used for innovation are widely distributed throughout the world. (See also Henry Chesbrough's article, "The Era of Open Innovation," p. 35.) The lessons of open-source projects demonstrate the value of specialization through self-selection and how norms of meritocracy and peer recognition help ensure product quality. To be sure, finding the right blend of incentives to encourage innovation is not an easy task. In the long run, however, managers may recognize that offering a mix of motives is the best way to encourage innovation; a mix that ranges from extrinsic and monetary-based incentives to the fulfillment of more-intrinsic needs, such as enhanced reputation among peers and community identification — that is, a sense of belonging.

Suggested Reading

Hackers: Heroes of the Computer Revolution, **Steven Levy** (New York: Anchor/Doubleday, 1984)

The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary, **Eric Raymond** (Sebastopol, California: O'Reilly, 1999)

Rebel Code: Linux and the Open Source Revolution, **G. Moody** (Cambridge, Massachusetts: Perseus Publishing, 2001)

Networks of Innovation, **I. Tuomi** (Oxford University Press, 2002)

"The Open Source Definition," B. Perens (1998), http://perens.com/Articles/OSD.html

"Why Open Source Software Can Succeed," A. Bonacorsi and C. Rossi *Research Policy* 32 (2003, in press)

"How Open Is Open Enough? Melding Proprietary and Open Source Platform Strategies," J. West *Research Policy* 32 (2003, in press)

"Institutional Entrepreneurship in the Sponsorship of Common Technological Standards: The Case of SUN Microsystems and Java," R. Garud, S. Jain and A. Kumaraswamy *Academy of Management Journal* 45, no. 1 (2002): 196-214

Reprint 4432

MITSloan
Management Review

Reprints/Back Issues

Electronic copies of SMR articles can be purchased on our website:
www.mit-smr.com

To order bulk copies of SMR reprints, or to request a free copy of our reprint index, contact:

MIT Sloan
Management Review
Reprints
E60-100
77 Massachusetts Avenue
Cambridge MA 02139-4307
Telephone: 617-253-7170
Toll-free in US or
Canada: 877-727-7170
Fax: 617-258-9739
E-mail: smr@mit.edu

Copyright Permission

To request permission to copy articles, contact:

P. Fitzpatrick,
Permissions Manager
Telephone: 617-258-7485
Fax: 617-258-9739
E-mail: pfitzpat@mit.edu